1. Pick the object

1. Pick the object, copypaste the query data into a text file for reference
2. pick at least a 10deg radius
3. pick time as long as possible for best results

Here's 3 clusters for this work

	cluster		
	Fornax	NGC4346	Virgo
RA	54.67	190.7	187.7
DEC	-35.31	2.688	12.34
MET time	239557417,582118346		
MeV range	1000,1000000		
Radius	10 degrees		

1 gtshit

1.1 pumping data together

First it's best to link all photon files thru a common text file

```
$ ls *PH* > binned_events.txt
```

1.2 gtselect

next it's best to select only the relevant data - we'll be aiming to do a binned analysis so we'll name the files accordingly anyway

```
$ gtselect evclass=128 evtype=3
Input file: @binned_events.txt
Output file: object_binned_filtered.fits
RA,DEC, Radius, start met, end met: INDEF
energy: match query
max zenith: 90
```

1.3 gtmktime

this step will make sure only the only data used is taken during times the telescope was working properly

```
$ gtmktime
spacecraft file: spacecraft file corresponding to the chosen time period
zenith cut: NO
event file: object_binned_filtered.fits
output event file: objects_binned_gti.fits
```

the selection can be verified wit the gtvcut:

```
$ gtvcut object_binned_gti.fits
extension name: EVENTS
```

1.4 gtbin - counts map

this will give a neat visualisation of the raw data and shit and can be used for later comparisons and shit as well as presentations and wowing others and crap

```
$ gtbin
output type: CMAP
event file: object_binned_gti.fits
output file: object_binned_cmap.fits
spacecraft: NONE
X,Y axis: diameter(in deg)/resolution(in pix/deg)
scale: see above
coords: see query
projection: AIT
```

the resulting cmap can be visualised with either fv5.5 or (better) ds9

1.5 gtbin - ccube

this will create a 3D binned cube counts map, with xy axes being ra dec as usual but z axis being energy
the binning of the counts map will determine the binning of the exposure calculation, this will take forever, so do it over the weekend or something, because it'll be using spacecraft data, so it's the time-log bins that will stretch it out, not the space-log bins
the cube has to fit in completely into the outscribed circle, a quick way to determine the side is radius*1.414, do this manipulation before decidin on the resolution

```
$ gtbin
output type: CCUBE
event file: object_binned_gti.fits
output file: object_binned_ccube.fits
spacecraft: NONE
X,Y axis: side(in deg)/resolution(in pix/deg)
scale: see above
coords: see query
projection: AIT
energy bins: LOG
energy range: see query
number of bins: idk, take something nice but not too stupid
```

1.6 XML sourcing

first thing first, wget/curl/copy the goddamn make4FGxml.py cos fuck doing this by hand. it'll take in a catalogue, binned events file and spit out a source file - by default it'll only spit out whatever htingy exsits, without any simulated power-law/dmfit/other sources, those have to be added manually later
the diffuse models are in $FERMI_DIR so echo that fucker to find out, because the python thing itself might mess up sometimes

```
$ python make4FGxml.py gll_psc_v18.fit object_binned_gti.fits -o object_input_model.xml
-G $FERMI_DIR/refdata/galdiffuse/gll_iem_v07.fits -g g11_iem_v07
-I $FERMI_DIR/refdata/galdiffuse/iso_P8R3_SOURCE_V2_v1.txt -i iso_P8R3_SOURCE_V2_v1
-s 120 -p TRUE
```

the psc file might need be downloaded cos reasons, the -s takes care of significance, -p includes point sources
the actual wannabe source should be added in manually (at the centre of the cluster or whatever) it's best to fit some stupid shit like powerlaw too for a comparison, and make sure to adjust the parameters accordingly (see query re: energy range, location)
but the actual source for DM will be dmfit - make sure to double check the path to the function, see the fermi website for deets on how to pick channels

1.7 gtltcube - livetimes and exposure

this is the fucking piece of shit that'll take forever, does what it says on the can, include a zenith cutoff to exclude rays reflected off earth:

```
gtltcube zmax=90
event file: object_binned_gti.fits
spacecraft: spacecraftfile
output file: object_binned_ltcube.fits
step size: 0.025
pixel size: 1
```

1.8 gtexpcube2 - exposure cube

here we'll calc the exposure map for the entire sky cos might as well, it'll improve shit, but also this is where the background map thing that'll be subracted is created. ofc we could just do a small portion that's just larger than our area of interest but fuck that, the time waiting difference between this and the entire sky ain't that much.
just make sure that the geometry is the same as for the ccube - ie make sure the degs/pixel is the same, and use 360/that thing and 180/that thing to get the values in pixel for the axes

```
$ gtexpcube2
livetime file: object_binned_ltcube.fits
counts map: none
output: object_binned_expcube.fits
response: P8R3_SOURCE_V2
size x,y, scale: see above
coordinates: see query
projection: AIT
energies: match ccube
```

1.9 gtsrcmaps - creating sourcemaps

here we'll make use of the fkn xml finally creating a file that will be passed on to the binned likelihood, it'll take all the point sources from the catalog as well as the diffusion ones and apply it onto the exposure map

```
$ gtsrcmaps
hypercube file: object_binned_ltcube.fits
counts file: object_binned_ccube.fits
exposure file: object_expcube.fits
source file: object_input_model.xml
response: CALDB
```

caveat: if this shit fails (especially file not found shit), instead of googling like mad, try to remake the input file a couple of times

1.10 gtlike - likelihood

fucking finally, alright, first of all, it's a fucking piece of shit running this on wsl, not because of performance, quite the contrary that's fine, but due to the fucking goddamn impossibility to display the fucking plot right away, so have fun with the output
~~oh brilliant idea - run xterm, that should work~~ nope, but clearly running this directly in lunex or windoze as opposed to wsl should be ok, maybe i'll come back to this later or at least figure out how to plot manually ffs
now, this is straight forward cos we have all the inputs:

```
$ gtlike refit=yes plot=yes sfile=object_binned_output.xml
results=object_results_type.dat specfile=object_spectra_type.fits [> object_log.txt]
stats: BINNED
counts file: object_srcmaps.fits
exposure file: object_expcube.fits
hypercube file: object_ltcube.fits
source file: object_input_model.xml
response: CALDB
optimizer: NEWMINUIT
```

not much to add, refit will offer a chance to edit the source and will attempt refitting, plot will plot (when possible), sfile is the output name, other shit gets logged into default names

1.11 gtmodel - creating a model map

now that the results and/or model source have been chucked into a binned output, we'll use that to create a new plot based off of that

```
$ gtmodel
source maps: object_srcmaps.fits
source model: object_binned_output.xml
output: object_model_map.fits
response: CALDB
exposure cube: object_ltcube.fits
binned exp map: object_expcube.fits
```

This will create a sweet af looking map to look at, but ofc it makes sense to compare this to the original. So we'll redo the entire restricted view thing again but with the original counts.

```
$ gtbin
output type: CMAP
event file: object_binned_gti.fits
output file: object_binned_cmap_small.fits
spacecraft: NONE
X, Y,axis: see ccube
scale: see ccube
coords: see query
projection: STG
```